ZX ASSEMBLER

USER MANUAL

SOFTWARE
for the

ZX81 • TS1000

CONTENTS **PAGE**

ZX ASSEMBLER

INTRODUCTION

The aim of ZX ASSEMBLER is to simplify the whole process of machine code programming on ZX/TS1000 computers. It provides comprehensive, easy to use facilities while retaining a surprisingly large amount of user-available RAM. The program occupies 7K and locates itself at the top of memory: thus BASIC is very nearly the same as without the ASSEMBLER present, enabling the use of assembly code subroutines within BASIC programs.

GENERAL DESCRIPTION

There are two main sections to the program:

(1) The **EDITOR/ASSEMBLER.** You may enter & edit your assembly code with the EDITOR then assemble it with the ASSEMBLE command. The EDITOR has been designed specially for the entering of assembly code: it employs up and down screen scrolling; search for a string; allows insertion/deletion of lines/characters; auto-repeat on all Keys; label indentation system & cursor up, down, left and right. It will allow up to 31 characters per line.

The ASSEMBLER is highly versatile: it provides for full mnemonic and label assembling, allows entry of numbers in hex and decimal, and it allows entry of text and comments.

(2) The **MONITOR.** This provides facilities to test, de-bug and run your machine-code program. It also provides a number of useful subroutines (such as Keyboard input; print on screen . . . etc.)

This manual is intended to explain how to operate the EDITOR/ASSEMBLER and MONITOR, also giving tips and routines for beginners, and a list of Z80 instruction set.

ZX ASSEMBLER is a powerful piece of software and used correctly could increase your programming speed and ability enormously.

GETTING STARTED

Read the relevant Chapter in your computer's Manual for Loading a program from a tape cassette.

The program is loaded using LOAD " " . The program takes approximately 3¼ minutes to LOAD.

Once the program has LOADED, it executes a NEW , and the K Cursor is displayed. To RUN the program you must type:

RAND USR 3E4

then the program will introduce itself. You are now in the COMMAND MODE of the ASSEMBLER. All commands are initiated with single Key press.

To return to BASIC, type Q (Quit) twice.

If you wish to continue editing an incomplete program:

Load the ASSEMBLER as described above. Now LOAD the incomplete program in the usual way. Then type:

RAND USR 3E4

If the program present leaves the ASSEMBLER with too little memory, error 4 is given when you type RAND USR 3E4.

SAVING PROGRAMS

To SAVE a complete or incomplete program, press Q twice to return to BASIC.

You will then find a two line BASIC program, plus any other that you have entered.

```
1 REM
2 REM
```

Line 1 contains the machine code.

Line 2 contains the assembly language.

You may wish at this point to delete either line as necessary and SAVE the remaining program in the usual way. The Assembler will not be saved. This program may be LOADED after the Assembler has been LOADED for further editing as necessary. (See Getting Started.)

If you have not finished your program, you can delete the object code (line 1) and SAVE only the source code.

However it is best to SAVE a backup tape containing only the source code for future ammendments.

THE EDITOR

When you start the EDITOR, the screen of text entered so far is displayed. This will be either at the place where you last left the EDITOR or from an error during Assembling. (See Assembling).

The EDITOR provides the user with several functions apart from simply entering text.

All the functions are obtained with the SHIFT Key held down.

FUNCTIONS

Press SHIFT 6	to move the cursor down (cursor only moves in the 7th character position).
Press SHIFT 7	to move the cursor up.
Press SHIFT 8	to move the cursor right.
Press SHIFT 5	to move the cursor left.
Press SHIFT 0	to RUBOUT the last character.
Press SHIFT 9	to INSERT a character at current position.
Press SHIFT A	to enter a LABEL. Moves cursor to extreme left of screen; only works if cursor is in 7th character position and if no label is there yet.
Press SHIFT D	DELETES current line.
Press SHIFT E	INSERTS a line at current cursor position.
Press SHIFT G	prints text from cursor position onto ZX Printer.
Press SHIFT Q	QUITS from Search, Line or Editor. Suppose you make an error in a line, SHIFT Q restores a line to its original state.
Press SHIFT S	SEARCH for a string from cursor position. Press SHIFT A to search for a label. If not found, cursor is left in same position.
Press SHIFT T	moves the cursor to the top of the text.

There is an AUTO REPEAT on all keys after 1 second.

The EDITOR allows a maximum of 31 characters per line. If you run out of memory, you will return to COMMAND MODE, and an OUT OF MEMORY message will appear: the line you have just typed will have been deleted so that you can return to the EDITOR to squeeze things up!

If a label is to be changed, SHIFT A won't move the cursor to the left, so use CURSOR LEFT.

SYNTAX of ASSEMBLER

Once the program has introduced itself, you have the choice of several commands for the monitor. We shall examine first the Editor/Assembler.

To enter the Editor, type E. The screen will clear and display a graphic space, seven characters in from the left hand side of the screen. You are now ready to enter your Assembly Language.

LABELS

The Assembler will accept labels of any string of letters or numbers starting with a letter immediately followed by:

= constant or

__ Assembly language __ - space

To enter a label type SHIFT 'A' and the cursor will move to the extreme left hand side of the screen. This gives the effect of indenting the Assembly language and making it easier to read.

Example: Score routine

```
SCORE1   =5000
         JR BEGIN          ; jump to beginning
SCORE    00 00             ; save 2 bytes for score
BEGIN    LD HL, (SCORE)    ; get the contents of score
         INC HL            ; increment it
         LD (SCORE), HL    ; store new score
         LD (SCORE), HL    ; store copy of score at 5000
                           HEX
         RET
```

A label may be accessed with an offset of −9 to +9.

```
e.g.     LD A, (PTR+4)
         JP 7A3C           ; Jump to MONITOR
PTR      1 2 3 4 5 6
```

A will have the value of 5. If no offset is specified, it is assumed it is zero. A=1.

N.B. It is advisable not to use a label with the same name as a register, e.g. BC or as a legal number, e.g. ABCD. This will give either an error message on relative jumps or potentialy disasterous consequences.

TEXT

Any text within quotation marks (" ") (SHIFT P) will be Assembled directly into HEX. This provides both an easy way of entering messages to be printed, and a way of using text in Assembly language commands.

Example: Print a message

```
        CAL 7E99              ; print text
        "HERE WE ENTER TEXT" FF
```

Example: Text in Assembly language

```
        LD A, O               ; get initial value
CHECK   INC A                 ; add 1 to it
        CP "*"                ; check for *
        JR NZ CHECK           ; if not go to check
        CALL 7EAA             ; print character
        CALL 7E99             ; print text
        " IS A * SIGN" FF
```

It is not necessary to know the character codes.

Any HEX Bytes typed in are also Assembled. So if you wish to save memory, you may type E5 C5 instead of PUSH HL PUSH BC.

NUMBERS

The Assembler defaults to HEXidecimal (base 16).

8 bit numbers 0 to FF

16 bit numbers 000 to FFFF (must be 3 or 4 digits)

Decimal numbers may also be used with a '+' or '−' sign. They will always be 16 bit numbers.

e.g. +1 is represented by 0001.

The Assembler is intelligent in may respects. It treats a number wherever possible, as an 8 bit or 16 bit, including decimal, when needed.

Example: label loads.

```
TEST    =+64
        LD HL, TEST        ; 16 bit expected
        LD A, TEST         ;  8 bit expected
```

The label TEST is treated as a 16 bit and 8 bit number where indicated.

If, however, the Assembler expects an 8 bit number when the number is greater than FF or +255, a NUMBER ERROR will be reported when Assembling.

Example: numbers

```
        LD HL,1            ; OK
        LD A,+13           ; OK
        LD A, 101          ; Error
```

When you wish to load a register with a number which is also a register, you must prefix the number with an 0.

Example

```
        LD A,B             ; stores register B in A
        LD A,0B            ; stores the number B in A
```

SEPARATORS

Each instruction may be separated by spaces or end of line.

Spaces may be replaced by multiple spaces with the exeption of the following instructions, which must be entered exactly as below.

LD__A,I	
LD__I,A	__-space
LD__R,A	
LD__A,R	
EX__AF	EX AF,AF'
EX__DE,HL	
EX__(SP), HL	

NON STANDARD MNEMONICS

The following Mnemonics are non standard Z80

EX__ EX AF, AF'
IM0 instead of IM__0
IM1 instead of IM__1
IM2 instead of IM__2

JP and JR have no commas. e.g. JR NZ+7

COMMENTS

Comments must be preceeded with a semicolon (;). Text after this will be ignored by the Assembler.

ASSEMBLING

Once you have typed in your Assembly language, you are ready to Assemble it. Exit from the Editor using SHIFT Q.

To assemble your program, type A. Then ASSEMBLE will be printed. Now press ENTER (N/L) and your program will be Assembled. The program is Assembled to start at address 4084 HEX or 16516 DECIMAL. This is REM line 1 when you return to BASIC.

If you make an error, an error message will be printed. Typing 'E' will display the text from the line at which the error occurred.

The error messages you may get are:

LABEL ERROR

(1) Undefined label.

(2) Illegal label definition. e.g. beginning with a number. If you misspell a Mnemonic, you will probably get this error. It thinks you are trying to access a label.

RELATIVE JUMP ERROR

This occurs if the relative jump is out of range, ie. the jump is $<+127$ or >-128 bytes.

OFFSET ERROR

(1) Loading a register with a label plus an offset.
e.g. LD HL, (FRED+10)
only ± 9 allowed

(2) Using the index registers plus an offset which is too large.
IX, IY $+127$
-128 allowed

NUMBER ERROR

This error is reported when you try to use a number too large for the relevant instruction.

e.g. LD A,1000

OUT OF MEMORY

(1) This error occurs, in Editor, when Assembly language is too long.
(2) During Assembling when there is not enough room to Assemble the program.

STACK ERROR

This occurs at the end of execution when too much has been taken or left on the stack. (See following notes.)

SYNTAX ERROR

Reports all other errors.

e.g. ADD A,*

While developing routines it is not necessary to put a RET at the end of your routine. The Assembler ends the routine with a Jump to the Monitor. This means if there is a Stack error, the system will not crash. The RET at the end of your program returns you to BASIC.

To RUN your routine, press 'R'. RUN FROM ADDR 4084 will be displayed. Press ENTER (N/L) to run your routine. The default address is 4084 (the beginning of the machine code routine). This may be changed by typing in any address.

NOTE: If an error is reported during assembly, the program cannot be RUN. It may only be partly assembled, or not at all!

MONITOR

ZX-ASSEMBLER also provides a useful Monitor to enable you to test, debug and run your machine code programs.

All commands are initiated with single Key-press. Many may need a 4 digit HEX address. Press ENTER (N/L) to enter this address. Any entry may be aborted by pressing Q before ENTER (N/L).

COMMENTS

A To ASSEMBLE your program

C to copy a block of memory from one place to another
COPY BLOCK:
FIRST ADDR . . . the first address of the block to be copied
LAST ADDR . . . the last address of the block to be copied
TO ADDR . . . the destination address

M This is the Memory EDIT mode. You can enter/edit M/C directly. Down the left side of the screen is a column of 24 2-byte addresses. Beside these is a column of the contents of each address. The cursor half-way down indicates the address of the memory location that you can change. To do this, type in a HEX number and press ENTER (N/L). If you press Q before ENTER (N/L), then the entry of that number will be aborted. The commands in this mode are:

- **J** calculates jump displacement for relative jumps: enter the last 2 digits of the destination address, press ENTER (N/L) and the displacement will be entered at the cursor address
- **L** moves cursor down continuously through memory (repeated ENTER)
- **ENTER** moves cursor down a single step
- **O** moves cursor up continuously
- **P** moves cursor up a single step
- **R** repeats the entry of a value: enter your value, then press ENTER (N/L). The contents of any address to which the cursor points will be changed to your value. Any of the cursor movement commands can be used. To cancel, repeat press Q.

I inspects and modify registers: displays register pairs BC, DE, HL and AF. F is also displayed in binary with each flat labelled (S=sign, Z=zero, H=half-carry, O=parity/overflow, N=negative or minus, C=carry). When you RUN a machine code program, the values displayed are put into the registers, and on return, the values of the registers are recorded. The HEX number you enter will be put into the register to which the cursor points. To enter a number and move the cursor press ENTER (N/L).

S SEARCH FOR . . . 2 byte number to be searched for
FROM ADDR . . . address from which to start search

Searches through all 64K except stack. Displays address of the first occurence of the number; if you press M, it goes straight into the Memory edit mode at that address. Press Q to return to COMMAND mode. 'NOT FOUND' is displayed if the number does not occur.

R Runs a machine code routine from address 4084 or the address you enter. Return to monitor is by a jump instruction assembled by the Assembler. The jump address is 7AC3.

The screen is cleared before the program is run, on return the Monitor waits for any key to be pressed before clearing the screen again and returning to COMMAND mode.

NOTE: Pressing Q (or SHIFT Q in the EDITOR) will always abort what you are doing whenever the Assembler is waiting for a key to be pressed.

USEFUL ASSEMBLER SUBROUTINES

The Assembler contains many subroutines which you may wish to use in your program. Here is a list of the more useful ones. The contents of all the registers (except F) are preserved unless otherwise stated.

Use the CALL instruction to call the subroutine within your program. The HEX address is given besides the subroutine name.

CLEAR SCREEN	- 7FB5	clears screen & homes screen cursor
HOME	- 7D7D	returns screen cursor to top left on screen
DISPLAY CHAR	- 7EAA	displays character code in register A: increments screens cursor (address 7FDO): cursor characters (70 - 73 HEX) can be used, screen automatically scrolls up or down as necessary
SCROLL UP	- 7F50	
SCROLL DOWN	- 7F6A	
DISPLAY TEXT	- 7E99	displays text following CALL instruction end with FF e.g. CALL 7E99 "HELLO" FF This will display HELLO at screen cursor.
HEX NUMBER	- 7E7E	displays register A in HEX
KEYBOARD	- 7EOF	scans keyboard once: returns charcter code in A; A=FF if no key was pressed
1 HEX IN	- 7DCO	returns the value in A register: D=FF if Q pressed
2 HEX IN	- 7D87	returns the value in HL: A is corrupted
CREATE REM	- 7B4E	creates a REM line in BASIC: DE Max addr that can be over written BC line number: HL length not including the EA 76
FIND LINE	- 7AEE	finds the address of a line in BASIC Input line number in BC Output address in HL
DELETE LINE	- 7B76	deletes a line in BASIC Input line number in BC

COPY BLOCK	- 7BEE	copies a block of memory as in the COPY command in the monitor 1st Address in BC beginning of block 2nd Address in DE end of block 3rd Address in HL new start address
NEWLINE	- 7F8A	causes an ENTER (N/L) (carriage return)
	- 7A3C	return address to Assembler

HINTS FOR THE BEGINNER

DON'T PANIC!

Machine Code is not a mysterious art practised by ice-cool whizz kids with electrons coursing through their veins. It is just another means of programming, like BASIC. You must be more careful than with BASIC as there are no error messages when you actually execute a program. If you make serious mistakes, the system crashes. However, by using the Assembler and Monitor and by following a few simple rules, you will soon be able to write, de-bug and successfully execute machine code.

RULES

1) Preserve your environment. Save all the registers within subroutines by pushing them on the stack. Before returning recover them in the opposite order to which they were saved.

```
e.g.    LD A,27          ; set A to a value
        PUSH HL          ; save HL
        PUSH BC          ; save BC
        CALL 7EAA        ; print character in A
        POP BC           ; recover BC
        POP HL           ; recover HL
```

REMEMBER with stacks, last ON, first OFF!

2) Everything placed on the stack, within a subroutine, must be taken off within the same routine, or before a RET is met.
Suppose FRED is a printing subroutine:

```
FRED LD A,17        ; set A to a value
     PUSH AF        ; save AF
     CALL 7EAA      ; print character
     RET            ; return?
```

This routine would crash as there should be a POP AF before the RET.

3) Remember where your stack is. Don't locate a machine code routine too close to the stack. This may cause the stack to corrupt the routine when it fills up.

4) Relative jumps have a limit from the current position
   +127 bytes forward
   −128 bytes backwards

5) Read as much information as you can about programming.

6) Put things in subroutines as much as reasonably possible.

RECOMMENDED BOOKS

Machine Language Programming Made Simple For Your Sinclair

Understanding Your ZX81 ROM

Programming the Z80 ZAKS

Z80 Instruction Handbook

ZX81 Cookbook

SAMPLE PROGRAMS

This program shows you how the galaxy may look from a space ship.

```
        LD BC, +7680      ; counter for number of
                            repeats
BEG1    CALL RND          ; get random number
        CP B
        JR NC STAR
        LD A,97           ; store code for *
        JR AGAIN
STAR    CP C
        JR NC BLAN
FREE    LD A,80           ; store code for graphic
                            space
        JR AGAIN
BLAN    CALL RND
        CP 7D
        JR C FREE
        LD A,9B           ; store code for graphic
AGAIN   CALL 7EAA         ; print stored character
        DEC BC            ; decrement counter
        LD A,B
        OR C              ; check if zero
        JR NZ BEG1        ; if not repeat
        RET
RND     LD A,R            ; put random number in A
        RES 7,A           ; make in range 0 - 128
        RET
```

This program slowly lands a space ship.

```
        LD DE,+700
LOOP    CALL 7E99
        ::++=" 72 72 72 72 FF; print space ship
        DEC DE
        LD B,0                  ; delay
        DJNZ -2
        LD A,D
        OR E                    ; count = 0
        JR NZ LOOP              ; if no, loop back
        CALL 7E99
        76 "** LANDED **" FF
```

MYSTERY PROGRAM

```
START   LD HL, START
        LD DE, NEXT+1
        LD B,5
LOOP    PUSH BC
        LD B,0
        LD A,(DE)
        LD C,A
        PUSH HL
        ADD HL,BC
        LD A,(HL)
        LD H, +57
        SCF
        CALL 7EAA
        LD L,0
        DEC HL
CONT    DEC HL      LD
        A,H     OR L
        JR Z E      SCF     JR
        CONT
        POP HL      POP BC
        INC DE
        DJNZ LOOP
        DEC D
NEXT    RL B
        LD (DE),A
        LD DE, 1C16
        JP 7A3C
```

This program displays message vertically in large letters.

```
        JR START
MES     "HI THERE THIS IS"      ; message
        " YOUR ZX81" FF         ; end with FF
TMP     00

START   LD DE,MES               ; DE is character pointer
NXTC    LD A,(DE)               ; get character
        CP FF                   ; is end?
        JR Z END
        AND 7F                  ; ignore inverse
        LD H,0 LD L,A
        ADD HL, HL              ; multiply by 8 to
        ADD HL,HL               ; give offset into
        ADD HL,HL               ; character table
        LD BC,1E00              ; add in base address
        ADD HL,BC               ; of character table
        LD B,8                  ; loop for character

NXTL    PUSH BC                 ; save count
        LD A,(HL)               ; get row in this character
        LD (TMP),A              ; store it at TMP
        LD B,8                  ; bit count of row

NXTB    LD A,(TMP)
        RLA
        LD (TMP),A              ; check bit
        LD A,(DE)
        JR C DISP               ; if set GOTO DISP
        BIT 7,A                 ; check for inverse
        JR Z 4
        LD A,80     JR 1        ; A=80 for inverse
XOR A   ; A=0 for non inverse

DISP    CALL 7EAA               ; print character
        DJNZ NXTB               ; next bit
        CALL 7E99     76 FF     ; display newline
        INC HL
        POP BC
        DJNZ NXTL               ; next row of character
        INC DE
        JR NXTC                 ; get next character

END                             ; of message
```

Z80 INSTRUCTION CODES

OBJ CODE	SOURCE STATEMENT	
8E	ADC	A,(HL)
DD8E05	ADC	A,(IX+d)
FD8E05	ADC	A,(IY+d)
8F	ADC	A,A
88	ADC	A,B
89	ADC	A,C
8A	ADC	A,D
8B	ADC	A,E
8C	ADC	A,H
8D	ADC	A,L
CE20	ADC	A,n
ED4A	ADC	HL,BC
ED5A	ADC	HL,DE
ED6A	ADC	HL,HL
ED7A	ADC	HL,SP
86	ADD	A,(HL)
DD8605	ADD	A,(1X+d)
FD8605	ADD	A,(1Y+d)
87	ADD	A,A
80	ADD	A,B
81	ADD	A,C
82	ADD	A,D
83	ADD	A,E
84	ADD	A,H
85	ADD	A,L
C620	ADD	A,n
09	ADD	HL,BC
19	ADD	HL,DE
29	ADD	HL,HL
39	ADD	HL,SP
DD09	ADD	1X,BC
DD19	ADD	1X,DE
DD29	ADD	1X,1X
DD39	ADD	1X,SP
FD09	ADD	1Y,BC
FD19	ADD	1Y,DE
FD29	ADD	1Y,1Y
FD39	ADD	1Y,SP
A6	AND	(HL)
DDA605	AND	(1X+d)
FDA605	AND	(1Y+d)
A7	AND	A
A0	AND	B
A1	AND	C
A2	AND	D
A3	AND	E
A4	AND	H
A5	AND	L

OBJ CODE	SOURCE STATEMENT	
E620	AND	n
CB46	BIT	0,(HL)
DDCB0546	BIT	0,(IX+d)
FDCB0546	BIT	0,(IY+d)
CB47	BIT	0,A
CB40	BIT	0,B
CB41	BIT	0,C
CB42	BIT	0,D
CB43	BIT	0,E
CB44	BIT	0,H
CB45	BIT	0,L
CB4E	BIT	1,(HL)
DDCB054E	BIT	1,(1X+d)
FDCB054E	BIT	1,(1Y+d)
CB4F	BIT	1,A
CB48	BIT	1,B
CB49	BIT	1,C
CB4A	BIT	1,D
CB4B	BIT	1,E
CB4C	BIT	1,H
CB4D	BIT	1,L
CB56	BIT	2,(HL)
DDCB0556	BIT	2,(1X+d)
FDCB0556	BIT	2,(1Y+d)
CB57	BIT	2,A
CB50	BIT	2,B
CB51	BIT	2,C
CB52	BIT	2,D
CB53	BIT	2,E
CB54	BIT	2,H
CB55	BIT	2,L
CB5E	BIT	3,(HL)
DDCB055E	BIT	3,(1X+d)
FDCB055E	BIT	3,(1Y+d)
CB5F	BIT	3,A
CB58	BIT	3,B
CB59	BIT	3,C
CB5A	BIT	3,D
CB5B	BIT	3,E
CB5C	BIT	3,H
CB5D	BIT	3,L
CB66	BIT	4,(HL)
DDCB0566	BIT	4,(1X+d)
FDCB0566	BIT	4,(1Y+d)
CB67	BIT	4,A
CB60	BIT	4,B
CB61	BIT	4,C
CB62	BIT	4,D

OBJ CODE	SOURCE STATEMENT	
CB63	BIT	4,E
CB64	BIT	4,H
CB65	BIT	4,L
CB6E	BIT	5,(HL)
DDCB056E	BIT	5,(1X+d)
FDCB056E	BIT	5,(1Y+d)
CB6F	BIT	5,A
CB68	BIT	5,B
CB69	BIT	5,C
CB6A	BIT	5,D
CB6B	BIT	5,E
CB6C	BIT	5,H
CB6D	BIT	5,L
CB76	BIT	6,(HL)
DDCB0576	BIT	6,(1X+d)
FDCB0576	BIT	6,(1Y+d)
CB77	BIT	6,A
CB70	BIT	6,B
CB71	BIT	6,C
CB72	BIT	6,D
CB73	BIT	6,E
CB74	BIT	6,H
CB75	BIT	6,L
CB7E	BIT	7,(HL)
DDCB057E	BIT	7,(1X+d)
FDCB057E	BIT	7,1Y+d)
CB7F	BIT	7,A
CB78	BIT	7,B
CB79	BIT	7,C
CB7A	BIT	7,D
CB7B	BIT	7,E
CB7C	BIT	7,H
CB7D	BIT	7,L
DC8405	CALL	C,nn
FC8405	CALL	M,nn
D48405	CALL	NC,nn
C48405	CALL	NZ,nn
F48405	CALL	P,nn
EC8405	CALL	PE,nn
E48405	CALL	PO,nn
CC8405	CALL	Z,nn
CD8405	CALL	nn
3F	CCF	
BE	CP	(HL)
DDBE05	CP	(1X+d)
FDBE05	CP	(1Y+d)
BF	CP	A
B8	CP	B
B9	CP	C
BA	CP	D
BB	CP	E
BC	CP	H
BD	CP	L
FE20	CP	n
EDA9	2C	
EDB9	CPDR	

OBJ CODE	SOURCE STATEMENT	
EDB1	CPIR	
EDA1	CPI	
2F	CPL	
27	DAA	
35	DEC	(HL)
DD3505	DEC	(1X+d)
FD3505	DEC	(1Y+d)
3D	DEC	A
05	DEC	B
0B	DEC	BC
0D	DEC	C
15	DEC	D
1B	DEC	DE
1D	DEC	E
25	DEC	H
2B	DEC	HL
DD2B	DEC	IX
FD2B	DEC	IY
2D	DEC	L
3B	DEC	SP
F3	DI	
102E	DJNZ	e
FB	EI	
E3	EX	(SP),HL
DDE3	EX	(SP),IX
FDE3	EX	(SP),IY
08	EX	AF
EB	EX	DE,HL
D9	EXX	
76	HALT	
ED46	IM	0
ED56	IM	1
ED5E	IM	2
ED78	IN	A,(C)
ED40	IN	B,(C)
ED48	IN	C,(C)
ED50	IN	D,(C)
ED58	IN	E,(C)
ED60	IN	H,(C)
ED68	IN	L,(C)
34	INC	(HL)
DD3405	INC	(1X+d)
FD3405	INC	(1Y+d)
3C	INC	A
04	INC	B
03	INC	BC
OC	INC	C
14	INC	D
13	INC	DE
1C	INC	E
24	INC	H
23	INC	HL
DD23	INC	IX
FD23	INC	IY
2C	INC	L
33	INC	SP
DB20	IN	A,(o)

OBJ CODE	SOURCE STATEMENT	
EDAA	IND	
EDBA	INDR	
EDA2	INI	
EDB2	INIR	
C38405	JP	nn
E9	JP	(HL)
DDE9	JP	(IX)
FDE9	JP	(IY)
DA8405	JP	C,nn
FA8405	JP	M,nn
D28405	JP	NC,nn
C28405	JP	NZ,nn
F28405	JP	P,nn
EA8405	JP	PE,nn
E28405	JP	PO,nn
CA8405	JP	Z,nn
382E	JO	C,e
302E	JR	NC,e
202E	JR	NZ,e
282E	JR	Z,e
182E	JR	e
02	LD	(BC),A
12	LD	(DE),A
77	LD	(HL),A
70	LD	(HL),B
71	LD	(HL),C
72	LD	(HL),D
73	LD	(HL),E
74	LD	(HL),H
75	LD	(HL),L
3620	LD	(HL),n
DD7705	LD	(1X+d),A
DD7005	LD	(1X+d),B
DD7105	LD	(1X+d),C
DD7205	LD	(1X+d),D
DD7305	LD	(1X+d),E
DD7405	LD	(1X+d),H
DD7505	LD	(1X+d),L
DD360520	LD	(1X+d),n
FD7705	LD	(1Y+d),A
FD7005	LD	(1Y+d),B
FD7105	LD	(1Y+d),C
FD7205	LD	(1Y+d),D
FD7305	LD	(1Y+d),E
FD7405	LD	(1Y+d),H
FD7505	LD	(1Y+d),L
FD360520	LD	(1Y+d),n
328405	LD	(nn),A
ED438405	LD	(nn),BC
ED538405	LD	(nn),DE
228405	LD	(nn),HL
DD228405	LD	(nn),IX
FD228405	LD	(nn),IY
ED738405	LD	(nn),SP
0A	LD	A,(BC)
1A	LD	A,(DE)
7E	LD	A,(HL)

OBJ CODE	SOURCE STATEMENT	
DD7E05	LD	A,(1X+d)
FD7E05	LD	A,(1Y+d)
3A8405	LD	A,(nn)
7F	LD	A,A
78	LD	A,B
79	LD	A,C
7A	LD	A,D
7B	LD	A,E
7C	LD	A,H
ED57	LD	A,I
7D	LD	A,L
3E20	LD	A,n
ED5F	LD	A,R
46	LD	B,(HL)
DD4605	LD	B,(1X+d)
FD4605	LD	B,(1Y+d)
47	LD	B,A
40	LD	B,B
41	LD	B,C
42	LD	B,D
43	LD	B,E
44	LD	B,H
45	LD	B,L
0620	LD	B,n
ED4B8405	LD	BC,(nn)
018405	LD	BC,nn
4E	LD	C,(HL)
DD4E05	LD	C,(1X+d)
FD4E05	LD	C,(1Y+d)
4F	LD	C,A
48	LD	C,B
49	LD	C,C
4A	LD	C,D
4B	LD	C,E
4C	LD	C,H
4D	LD	C,L
0E20	LD	C,n
56	LD	D,(HL)
DD6605	LD	D,(1X+d)
FD5605	LD	D,(1Y+d)
57	LD	D,A
50	LD	D,B
51	LD	D,C
52	LD	D,D
53	LD	D,E
54	LD	D,H
55	LD	D,L
1620	LD	D,n
ED5B8405	LD	DE,(nn)
118405	LD	DE,nn
5E	LD	E,(HL)
DD5E05	LD	E,(1X+d)
FD5E05	LD	E,(1Y+d)
5F	LD	E,A
58	LD	E,B
59	LD	E,C
5A	LD	E,D

OBJ CODE	SOURCE STATEMENT	
5B	LD	E,E
5C	LD	E,H
5D	LD	E.L
1E20	LD	E,n
66	LD	H,(HL)
DD6605	LD	H,(1X+d)
ED6605	LD	H,(1Y+d)
67	LD	H,A
60	LD	H,B
61	LD	H,C
62	LD	H,D
63	HD	H,E
64	LD	H,H
65	LD	H,L
2620	LD	H,n
2A8405	LD	HL,(nn)
218405	LD	HL,nn
ED47	LD	1A
DD2A8405	LD	1X,(nn)
DD218405	LD	1X,nn
FD2A8405	LD	1X,(nn)
FD218405	LD	1Y,nn
6E	LD	L,HL
DD6605	LD	L,(1X+d)
FD6605	LD	L,(1Y+d)
6F	LD	L,A
68	LD	L,B
69	LD	L,C
6A	LD	L,D
6B	LD	L,E
6C	LD	L,H
6D	LD	L,L
2E20	LD	L,n
ED4F	LD	R,A
EDDB8405	LD	SP,(nn)
F9	LD	SP,(HL)
DD69	LD	SP,IX
FD69	LD	SP,IY
316405	LD	SP,nn
EDAB	LDD	
EDb8	LDDR	
EDAC	LD1	
EDBC	LD,R	
ED44	NEG	
00	NDP	
B6	OR	HL
DDB605	OR	(1X+d)
FDB606	OR	(1Y+d)
B7	OR	A
B0	OR	B
B1	OR	C
B3	OR	E
B4	OR	H
B5	OR	L
F620	OR	n
ED88	ORDR	

OBJ CODE	SOURCE STATEMENT	
EDB3	OUR	
EDB9	OUT	C,A
ED41	OUT	C,B
ED49	OUT	C,C
ED51	OUT	C,D
ED59	OUT	C,E
ED61	OUT	C,H
ED69	OUT	C,L
D32C	OUT	A
EDAB	OUTD	
EDA3	OUTE	
F1	POP	AB
C1	POPBC	
D1	POP	DE
E1	POP	HL
DDE1	POP	1X
FDE1	POP	1Y
F5	PUSH	AF
C5	PUSH	BC
D5	PUSH	DE
E5	PUSH	HL
DDE5	PUSH	IX
FDE5	PUSH	IY
CB86	RES	O,(HL)
DDCB058E	RES	0,(1X+d)
FDCB0686	RES	0,(1Y+d)
CB87	RES	0,A
CB80	RES	0,B
CB81	RES	0,C
CB82	RES	0,D
CB83	RES	0,E
CB84	RES	0,H
CB85	RES	0,L
CB8E	RES	1,(HL)
DDCB058E	RES	1,(1X+d)
FDCB058E	RES	1,(1Y+d)
CB8F	RES	1,A
CB88	RES	1,B
CB89	RES	1,C
CB8A	RES	1,D
CB8B	RES	1,E
CB8C	RES	1,H
CB8D	RES	1,L
CB96	RES	2,HL
DDCB0696	RES	2,(1X+d)
FDCB0696	RES	2,(1Y+d)
CB97	RES	2,A
CB90	RES	2,B
CB91	RES	2,C
CB92	RES	2,D
CB93	RES	2,E
CB94	RES	2,F
CB95	RES	2,G
CB9E	RES	2,H
DDCB069E	RES	3,(1X+d)
FDCB069E	RES	3,(1Y+d)

OBJ CODE	SOURCE STATEMENT	
CB9F	RES	3,A
CB98	RES	3,B
CB99	RES	3,C
CB9A	RES	3,D
CB9B	RES	3,E
CB9C	RES	3,H
CB9D	RES	3,L
CBA6	RES	4,(HL)
DDCB05A6	RES	4,(1X+d)
FDCB05A6	RES	4,(1Y+d)
CBA7	RES	4,A
CBA0	RES	4,B
CBA1	RES	4,C
CBA2	RES	4,D
DBA3	RES	4,E
CBA4	RES	4,H
CBA5	RES	4,L
CBAE	RES	5,(HL)
DDCB05AE	RES	5,(1X+d)
FDCB05AE	RES	5,(1Y+d)
CBAF	RES	5,A
CBA8	RES	5,B
CBA9	RES	5,C
CBAA	RES	5,D
CBAB	RES	5,E
CBAC	RES	5,H
CBAD	RES	5,L
CBB6	RES	6,(HL)
DDCB05B6	RES	6,(1X+d)
FDCB05B6	RES	6,(1Y+d)
CBB7	RES	6,A
CBB0	RES	6,B
CBB1	RES	6,C
CBB2	RES	6,D
CBB3	RES	6,E
CBB4	RES	6,H
CBB5	RES	6,L
CBBE	RES	7,(HL)
DDCB05BE	RES	7,(1X+d)
FDCB05BE	RES	7,(1Y+d)
CBBF	RES	7,A
CBB8	RES	7,B
CBB9	RES	7,C
CBBA	RES	7,D
CBBB	RES	7,E
CBBC	RES	7,H
CBBD	RES	7,L
C9	RET	
D8	RET	C
F8	RET	M
DO	RET	NC
CO	RET	NZ
FO	RET	P
E8	RET	PE
EO	RET	PO
C8	RET	Z

OBJ CODE	SOURCE STATEMENT	
ED4D	RET1	
ED45	RETN	
CB16	RL	(HL)
DDCB0516	RL	(1X+d)
FDCB0516	RL	(1Y+d)
CB17	RL	A
CB10	RL	B
CB11	RL	C
CB12	RL	D
CB13	RL	E
CB14	RL	H
CB15	RL	L
17	PLA	
CB06	RLC	(HL)
DDCB0506	RLC	(1X+d)
FDCB0506	RLC	(1Y+d)
CB07	RLC	A
CB00	RLC	B
CB01	RLC	C
CB02	RLC	D
CB03	RLC	E
CB04	RLC	H
CB05	RLC	L
07	RLCA	
ED6F	RLD	
CB1E	RR	(HL)
DDCB051E	RR	(1X+d)
FDCB051E	RR	(1Y+d)
CB1F	RR	A
CB18	RR	B
CB19	RR	C
CB1A	RR	D
CB1B	RR	E
CB1C	RR	H
CB1D	RR	L
1F	RRA	
CBDE	RRC	(HL)
DDCB050E	RRC	(1X+d)
FDCB050E	RRC	(1Y+d)
CB0F	RRC	A
CB08	RRC	B
CB09	RRC	C
CB0A	RRC	D
CB0B	RRC	E
CB0C	RRC	H
CB0D	RRC	L
OF	RRCA	
ED67	RRD	
C7	RST	00H
CF	RST	08H
D7	RST	10H
DF	RST	18H
E7	RST	20H
EF	RST	28H
F7	RST	30H
FF	RST	38H
DE20	SBC	A,n

OBJ CODE	SOURCE STATEMENT	
9E	SBC	A,(HL)
DD9E05	SBC	A,(1X+d)
FD9E05	SBC	A,(1Y+d)
9F	SBC	A,A
98	SBC	A,B
99	SBC	A,C
9A	SBC	A,D
9B	SBC	A,E
9C	SBC	A,H
9D	SBC	A,L
ED42	SBC	HL,BC
ED52	SBC	HL,DE
ED62	SBC	HL,HL
ED72	SBC	HL,SP
37	SCF	
CBC6	SET	0,(HL)
DDCB05C6	SET	0,(1X+d)
FDCB05C6	SET	0,(1Y+d)
CBC7	SET	0,A
CBCO	SET	0,B
CBC1	SET	0,C
CBC2	SET	0,D
CBC3	SET	0,E
CBC4	SET	0,H
CBC5	SET	0,L
CBCE	SET	1,(HL)
DDCB05CE	SET	1,(1X+d)
FDCB05CE	SET	1,(1Y+d)
CBCF	SET	1,A
CBC8	SET	1,B
CBC9	SET	1,C
CBCA	SET	1,D
CBCB	SET	1,E
CBCC	SET	1,H
CBCD	SET	1,L
CBD6	SET	2,(HL)
DDCB05D6	SET	2,(1X+d)
FDCB05D6	SET	2,(1Y+d)
CBD7	SET	2,A
CBD0	SET	2,B
CBD1	SET	2,C
CBD2	SET	2,D
CBD3	SET	2,E
CBD4	SET	2,H
CBD5	SET	2,L
CBD8	SET	3,B
CBDE	SET	3,(HL)
DDCB05DE	SET	3,(1X+d)
FDCB05DE	SET	3,(1Y+d)
CBDF	SET	3,A
CBD9	SET	3,C
CBDA	SET	3,D
CBDB	SET	3,E
CBDC	SET	3,H
CBDD	SET	3,L
CBE6	SET	4,(HL)

OBJ CODE	SOURCE STATEMENT	
DDCB05E6	SET	4,(1X+d)
FDCB05E6	SET	4,(1Y+d)
CBE7	SET	4,A
CBEO	SET	4,B
CBE1	SET	4,C
CBE2	SET	4,0
CBE3	SET	4,E
CBE4	SET	4,H
CBE5	SET	4,L
CBEE	SET	5,(HL)
DDCB05EE	SET	5,(1X+d)
FDCB05EE	SET	5,(1Y+d)
CBEF	SET	5,A
CBE8	SET	5,B
CBE9	SET	5,C
CBEA	SET	5,D
CBEB	SET	5,E
CBEC	SET	5,H
CBED	SET	5,L
CBF6	SET	6,(HL)
DDCB05F6	SET	(1X+d)
FDCB05F6	SET	6,(1Y+d)
CBF7	SET	6,A
CBF0	SET	6,B
CBF1	SET	6,C
CBF2	SET	6,D
CBF3	SET	6,E
CBF4	SET	6,H
CBF5	SET	6,L
CBFE	SET	7,(HL)
DDCB05FE	SET	7,(1X+d)
FDCB05FE	SET	7,(1Y+d)
CBFF	SET	7,A
CBF8	SET	7,B
CBF9	SET	7,C
CBFA	SET	7,D
CBFB	SET	7,E
CBFC	SET	7,H
CBFD	SET	7,L
CB26	SLA	(HL)
DDCB0526	SLA	(1X+d)
FDCB0526	SLA	(1Y+d)
CB27	SLA	A
CB20	SLA	B
CB21	SLA	C
CB22	SLA	D
CB23	SLA	E
CB24	SLA	H
CB25	SLA	L
CB2E	SRA	HL
DDCB052E	SRA	(1X+d)
FDCB052E	SRA	(1Y+d)
CB2F	SRA	A
CB28	SRA	B
CB29	SRA	C
CB2A	SRA	D

OBJ CODE	SOURCE STATEMENT	
CB2B	SRA	E
CB2C	SRA	H
CB2D	SRA	L
CB3E	SRA	(HL)
DDCB053E	SRL	(1X+d)
FDCB053E	SRL	(1Y+d)
CB3F	SRL	A
CB38	SRL	B
CB39	SRL	C
CB3A	SRL	D
CB3B	SRL	E
CB3C	SRL	H
CB3D	SRL	L
96	SUB	(HL)
DD9605	SUB	(1X+d)
FD9605	SUB	(1Y+d)
97	SUB	A
90	SUB	B
91	SUB	C
92	SUB	D
93	SUB	E
94	SUB	H
95	SUB	L
D620	SUB	n
AE	XOR	(HL)
DDAE05	XOR	(1X+d)
FDAE05	XOR	(1Y+d)
AF	XOR	A
A8	XOR	B
A9	XOR	C
AA	XOR	D
AB	XOR	E
AC	XOR	H
AD	XOR	L
EE20	XOR	n

Courtesy of Zilog Inc.

ZX ASSEMBLER

The aim of ZX ASSEMBLER is to simplify the whole process of machine code programming on ZX/TS1000 computers. It provides comprehensive, easy to use facilities while retaining a surprisingly large amount of user-available RAM. The program occupies 7K and locates itself at the top of memory: thus BASIC is very nearly the same as without the ASSEMBLER present, enabling the use of assembly code subroutines within BASIC programs.

ASK YOUR DEALER ABOUT THESE FINE PROGRAMS

• GAMES • GAMES • GAMES •

MARINE RESCUE: fast action underwater
TANK TRAP: destroy enemy planes & land mines
GALAXY INVADERS: repel fleets of invaders
INVADERS: the classic computer game
ZX SCRAMBLE: a fast-moving space game
CROWN & SCEPTER: a medieval adventure
GALACTIC COMMANDO: a space war adventure
TRACK DOWN: an adventure in the old west
BLACKJACK: as played in Vegas
SLOTS: beat the one-armed bandits
ZX CHESS I: 6 levels: black or white: save games
ZX CHESS II: chess master: 7 levels: champion rated
1K CHESS: 1 level: no castling or en-passant
PLANET OF FEAR: find your stolen spaceship
INCA CURSE: get gold out of the temple
SHIP OF DEATH: free your ship from an alien cruiser
NANTIR RAIDERS: arcade game: 4 waves of attackers
GOBBLE MAN: famous arcade game: chase ghosts in a maze
1K GAMES: 11 games for unexpanded ZX81/TS1000
SHOOT OUT: how fast are you "on the draw"
BATTLE SHIPS: the classic naval war game
POP STAR: enter the exciting world of popular music
KEYS TO GONDRUN: move through a kingdom of magical adventure
INHERITANCE: prove your financial wizardry and inherit your Great Uncle's estate
AROUND EUROPE IN 80 HOURS: race through Europe in under 80 hours to win
RUN THE COUNTRY: you be the chief executive in your own country
ZOMBIES & SWORD OF PEACE: play these two exciting games on one tape

FAMILY EDUCATION & HEALTH & ENTERTAINMENT

WEIGHT CONTROL: a personalized weight loss program*
CONSTELLATION: your computer is your telescope*
SOLAR SYSTEM FILE: a databank on the solar system*
BIORHYTHMS: plot your physical, emotional, intellectual cycles*
BOOK OF DAYS: facts, trivia, birthdays in a datafile*
PERSONAL RECORDS STORAGE SYSTEM: create a personal datafile*
FLASHCARD: memory aid, learning aid, testing device
MOVIE HANGMAN: guess the movie; beat the Hangman

PROGRAMMING AIDS

Z-AID 1.0: a machine code programming aid
ZX BUG: for debugging, editing & running machine codes
ZX ASSEMBLER: powerful tool for machine code programs
TOOLKIT: add 9 commands to basic; including renumber
ZX FORTH: ease of basic with machine code speed*
FASTLOAD: load any ZX81/TS1000 program 4 or 6 times faster than normal

** comes complete with a detailed guide*

INTERNATIONAL PUBLISHING & SOFTWARE INC.

3952 Chesswood Drive, Downsview, Ontario, Canada M3J 2W6